Harsh Kumar, 24576, **Edge AI Assignment –1, 08/02/2025**

1. How many samples were you able to collect per second?

Using the Arduino Nano BLE Sense board, I was able to collect IMU sensor data at a rate of around 1-2 samples per second. The speed was mainly limited by the calculations performed and the communication rate. While the program achieved a sample rate of 4-5 Hz in the Arduino MicroPython IDE, when using communication, the average rate dropped to approximately 1.32 Hz.

2. What changes would you make to your code to increase the sampling rate?

- Replace the list-based queue (using `pop(0)`) with a circular buffer or deque to lower processing overhead.
- Increase the UART baud rate to reduce transmission delay and to share less data in 1 iteration.

3. What challenges did you encounter while collecting real-world data using IMU sensors?

- Raw IMU data (from accelerometer, gyroscope, magnetometer) is noisy and shows drift, requiring additional filtering and calibration.
- Real-time feature extraction on a sliding window is intensive on MicroPython, creating a bottleneck in faster data collection.
- The large number of fields (992 per sample) and low baud rate create delays in data transmission.
-

4. Any other insights or observations?

- There is a balance between detailed feature extraction and maintaining a high sample rate.
- With large file sizes (e.g. 5,000 samples per activity), proper organisation (identifiers) is crucial to avoid data loss.
- Have few communication errors within the data, leading small data loss.